Interfaces for Wide Area Information Servers

1/17/92 Version 0.8

Brewster Kahle, Harry Morris, Jonathan Goldman
Thinking Machines Corporation

Thomas Erickson
Apple Computer

John Curran
NSF Network Service Center

Email addresses: Brewster@think.com, morris@think.com, jonathan@think.com, thomas@apple.com, jcurran@nnsc.nsf.net

ABSTRACT

Interfaces for information access and retrieval are a long way from the ideal of the electronic book that you can cuddle up with in bed. Nevertheless, today's interfaces are coming closer to supporting browsing, selection, and retrieval of remote information by non-technical users.

The Wide Area Information Server (WAIS) system is built on a standard protocol, and thus supports access to a wide range of remote databases. As a consequence, it is a good platform on which to explore the design of interfaces for information retrieval. This paper describes 5 interfaces to WAIS that have been designed and implemented: WAIStation for the Macintosh, Rosebud for the Macintosh, XWAIS for X Windows, GWAIS for Gnu-Emacs, and SWAIS for dumb terminals.

The interfaces to WAIS described here reflect a variety of design constraints. Such constraints range from the mundane—coping with dumb terminals and limited screen space—to the challenging. Among the challenges addressed are how to provide passive alerts, how to make information easily scannable, and how to support retrieval and browsing by non-technical users. There are a variety of other issues which have received little or no attention, including budgeting money for access to 'for pay' databases, privacy, and how to assist users in finding out which of a large (changing) set of databases holds relevant information. We hope that the challenges we have identified, as well as the existence and public availability of source code for the WAIS system, will serve as a stimulus for further design work on interfaces for information retrieval.

1. INTRODUCTION

It requires little prescience to predict that one day computers will put an ocean of information at the finger tips of a vast population of users. However, although there is a considerable amount of information available from remote sources, the bulk of it is accessible only to information professionals, or users with highly technical backgrounds. A variety of obstacles effectively block the ordinary user from accessing information via the computer. These obstacles include the difficulty of locating appropriate information sources, the cumbersome maneuvers needed to get on-line and to connect to remote sources, and cryptic query languages. Furthermore, even if a user has succeeded in accessing a remote information source, it is likely that it will have its own special purpose interface, which may or may not support the user's needs.

In this paper we describe the Wide Area Information Servers (WAIS) project, which provides a protocol-based mechanism for accessing a variety of remote, full text information servers. WAIS has the potential for supporting a single interface to a wide variety of information sources, and offers a good platform on which to explore the design of interfaces for information retrieval. After a summary of existing information retrieval systems, we describe the WAIS system, and then describe the 5 interfaces which have been designed for it. In the course of these descriptions we discuss design constraints, interface issues, and practical matters which impacted the designs. We conclude with a summary, and some remarks on important issues which have not been addressed, and a invitation for other investigators to use the WAIS system as a platform for exploring interfaces to multiple, remote information sources.

2. BACKGROUND

2.1 Existing Systems

While a review of all existing systems is beyond the scope of this paper, it is useful to list a number of the most popular or significant interfaces for information retrieval.

Commercial interfaces for accessing full text resources on computers can be broken down into dialup services, local file access, and LAN-based access tools. Dialup systems such as Dialog and Dow Jones offer TTY interfaces to users, with menus and command lines being the dominant access tools. Some dialup services are offering client programs that run on personal computers to add graphical interfaces such as "Navigator" by Compuserve. In general, these interfaces are unique to the information provider. Local file access through full-text indexing has been achieved in command line form (e.g. the unix command "grep") and in screen based interfaces (e.g. ON Location (ON), and Digital Librarian (NeXT)). These interfaces often give browsing and searching capabilities for local files. Some of these interfaces have been stretched to work with files on file servers. LAN-based access tools usually use some sort of query language to access servers on the net, such as Verity's Topic system (VERITY), and numerous library systems. These query languages require some user training. Integrated tools for cross platform, cross vendor information access are not currently available in other systems.

A variety of research projects have explored information retrieval systems. The SuperBook project (Egan, 1989) targets users of static information. Project Mercury (Ginther-Webster, 1990) is a remote library searching system that uses a client-server model. Information Lens (Malone, 1986) is a structured email system for assisting in managing corporate information. NetLib for software (Dongarra, 1987) and Mosis for information on how to fabricate chips (Mosis) are examples of email based information retrieval systems.

2.2 Overview of the WAIS Project

The ultimate goal of WAIS was to define an open protocol which would allow any user interface or information server that talked the protocol to interact with any other component which used the protocol. From the user's perspective, this would mean that user interfaces and information sources could be mixed and matched, according to the user's needs.

WAIS started as a joint project between Thinking Machines Corporation, Apple Computer, Dow Jones & Co., and KPMG Peat Marwick (Kahle, 1991a). The proximate goal was to define the open protocol and demonstrate its feasibility by implementing and demonstrating a multi-vendor system which provided ordinary users with access to a variety of remote databases. Thinking Machines contributed its Connection Machine based retrieval technology,

Apple contributed its user interface expertise, and Dow Jones & Co. provided access to its commercial information sources. KPMG Peat Marwick provided access to its corporate data, and served as a test site. The resulting system was installed at KPMG Peat Marwick and enabled the designers to study the success of the system in a real world context. The existing system uses pseudo natural language queries, relevance feedback to refine queries, and accesses full text, unstructured information sources. These technologies were used because they had already been tested independently, thereby leading to faster implementation of the complete system.

After the collaborative phase of the WAIS project came the Internet experiment. In this phase of WAIS, source code for the open protocol, information servers, and for several interfaces were made freely available over the internet. In addition, Thinking Machines established and maintained a directory of information servers, which WAIS users could query to find out about available information sources. This phase of WAIS is still in progress, and has resulted in the creation of new interfaces, the availability over the internet of more than a hundred servers on three continents, and over 100,000 searches of the directory of servers. In the first 6 months of the Internet experiment, approximately 4000 users from 20 countries have tried this system, with no training other than documentation (Kahle, 1991b). Administrators of popular information servers indicate that they are getting over 50 accesses a day from many countries.

2.3 The WAIS System

WAIS employs a client-server model using a standard protocol (based on Z39.50) to allow users to find and retrieve information from a large number of servers. The client program is the user interface, the server does the indexing and retrieval of documents, and the protocol is used to transmit the queries and responses. Any client which is capable of translating a user's request into the standard protocol can be used in the system. Likewise, any server capable of answering a request encoded in the protocol can be used.

A WAIS server can be located anywhere that one's workstation has access to: on the local machine, on a network, or on the other end of a modem. The user's workstation keeps track of a variety of information about each server. The public information about a server includes how to contact it, a description of the contents, and the access cost.

The WAIS protocol (Davis, 1990) is an extension of the existing Z39.50 standard (NISO, 1988) from NISO. It has been augmented where necessary to incorporate many of the needs of a full-text information retrieval system. To allow future flexibility, the standard does not restrict the query language or the data format of the information to be retrieved. Nonetheless, a query convention has been established for the existing servers and clients. The resulting WAIS Protocol is general enough to be implemented on a variety of communications systems.

The WAIS clients will be described in detail in the next several sections. However, all of them work in a basically similar way. On the client side, queries are expressed as strings of words, often pseudo natural language questions. The client application then packages the query in the WAIS protocol, and transmits it over a network to one or more servers. The servers receive the transmission, translate the received packet into their own query languages, and search for documents satisfying the query. The lists of relevant documents are then encoded in the protocol, and transmitted back to the client. The client decodes the response, and displays the results. The documents can then be retrieved from the server. The documents can be in any format that the client can display such as word processor files or pictures.

3. WAISTATION: AN INTERACTIVE QUERY INTERFACE

WAIStation At A Glance	
Target Machine	Macintosh Plus and above, 9" Monochrome screen.
Effort	1 man-year
Number of Users	2000
Status	finished, freely distributed
Language	ThinkC
Communications	TCP/IP and Modem (not supported)
Designer	Harry Morris
Organization	Thinking Machines
Availability	Available for anonymous FTP from /public/wais/WAIStation*.sit.hqx@think.com
Design goals	Implementable quickly, support interactive queries well, changeable based on user's comments, make something very simple to learn (partner friendly), try out many ideas: interactive queries, passive alerting, asking multiple servers.
Used	In a study with accountants and tax consultants at KPMG: very good user acceptance. In the Internet experiment: estimated that half of the uses of WAIS are using WAIStation. (based on when the directory of servers did not work for Macintoshes, usage dropped to half).
Problems	dealing with the directory of servers (s). Modem code was difficult to get right.

WAIStation was designed for use in the WAIS experiment at KPMG Peat Marwick. As such, we needed an interface that would be easy to use, and would encourage successful searches by users untrained in search techniques. Peat Marwick often sends its employees into the field toting their Macintosh SE's along for use as portable computers. Thus we had to design the interface to run on a 9-inch black-and-white screen, and make minimal demands on CPU and memory. Furthermore, WAIStation was designed for use over modems and slow LANs.

3.1 Design Rationale

In designing WAIStation, we were informed by two metaphors - search as conversation, and storage by file folder. The process of formulating an effective search is highly interactive. Of the documents which match a query, the ones which match "best" are displayed. One or more may be of interest, in which case, they can be fed back to the system, interactively improving the search. We choose to view this process as a conversation. Thus the initial natural language question becomes that starting point for give and take between the user and the server(s). Relevance feedback provides the context for the question. As the search proceeds, some results may suggest alternative searches or branches of the conversation. This is provided for by allowing several questions to evolve at the same time.

Eventually one or more questions may be refined to the point where they are finding consistently good results. At this point, the question can be automated, becoming a dynamically updated file folder. At intervals these questions wake up and query their servers. The results are stored in the results field for later inspection. They can be thought of as regular Macintosh folders, except augmented with a charter describing how to keep their contents up to date.

This parallel with the Macintosh folder structure suggested a drag and drop construction for the user interface itself. Constructing a question is a three step process - typing the key words, specifying the servers to use, and specifying the relevant documents to feed back. If we think of questions like Macintosh folders, we can use the Macintosh's drag and drop mechanism for putting sources and relevant documents into a question. This approach makes WAIStation's mechanics instantly familiar to users of the Macintosh finder.

3.2 Human Interface

When WAIStation starts up, two windows appear – one contains the users available Sources (see below) and one contains the users saved Questions. Sources are identified by an eye icon, questions by a question mark icon.

Double clicking on a question icon opens the stored question, including any new results found since the last time it was examined. The top half of the question window contains a field in which to type key words (the natural language part of the question), a list of relevant documents, a list of sources, and a list of result headlines. Sources can be added to the question buy selecting a source icon (in the Sources window), and dragging it into the question. Relevant documents are specified in the same way.

Result documents, returned by the servers, can be examined by double clicking on their icon. Note that the result list contains a graphical indication of how well each document matches the query. The original graphic was a series of 0 to 4 stars, similar to the ratings found in TV guide. We thought that this rating scheme would be easily recognized. Experience proved that the stars did not provide enough information to be recognized, or to discriminate among the documents. Latter versions of the software replaced the stars with a horizontal bar giving 20 levels of resolution.

Any of the resulting documents can be opened and viewed in its own window. WAIStation supports plain ascii documents as well as PICT format pictures. Text windows automatically scroll to the position which the server considers the most relevant part of the document. This allows the user to quickly determine if a file is useful. In order to perform well over slow communications channels (modems and slow LANs) the text is downloaded on demand in 15 line chunks. The keywords used in the query are automatically highlighted in bold.

Sources are specially formatted text files which describe information servers and how to get to them. Double clicking on a source displays a window with several controls. The top part is information specified by the server itself - a pop-up menu to specify the method of contacting the server (ip-address/tcp-port, modem number and speed, or location of a local index); a script to run after logging in (for use by modems); a database to search (servers can support multiple databases); a display of when the server is updated, how much it costs to search, and a textual description of the databases' contents. The bottom half of the source window allows the user to specify personal information about the server - when to contact it (for automatic update); when it was last contacted; how much to spend on it; how much credence its results should be given (this is used to scale document scores, which helps in the sorting of responses to questions asked of multiple servers); the number of documents to ask for when searching it; and finally the font and type size to use when displaying plain text results (important to publishers). Several of these fields are merely place holders in the current implementation. In particular, budget and confidence have not been implemented yet since there are no for-pay servers yet, and the number of sources is still relatively small.

Source files can also be retrieved from servers. This allows users to search servers whose database elements are pointers to other servers. The results can be used as targets for further searches. An experimental directory of servers is being maintained on the Internet.

3.3 Implementation

WAIStation was implemented in Think C 4.0 using the object oriented class library. It took about a man year of effort. The most difficult parts were the automatic update facility and the communications. Automatic Update required the ability to do background processing - which is not a normal part of the Macintosh operating system. Communications were difficult primarily because we were simultaneously debugging the Z39.50 protocol, modem code, and the (then new) Apple Communications Toolbox. We eventually left modems unsupported, and replaced the Communications Toolbox with direct calls to MacTCP. Through this experience we found that communications speeds of less than 9600 baud were barely tolerable for interactive text retrieval.

3.4 Observations

We estimate that WAIStation is now in use by over 2000 users in twenty countries. The common user complaints center around configuring MacTCP, using (the undocumented) directory-of-servers, and avoiding a bug requiring the software to be installed on the start up disk.

We have noticed several shortcomings in the current design:

- Users want access to their own data - WAIStation is capable of searching a Macintosh based inverted index file, but we unbundled the index builder when we realized how much work it would take to make it useful under Macintosh OS. OnLocation (On Technology) is an implementation of a Macintosh indexer that could be used.
- Interaction with the directory of servers is incomplete - It is not obvious which search results are source files, and what to do with the ones that are. It should be possible to drag a retrieved source directly into a question's source window, but the present interface requires that it be saved first. The lesson we learned was that special cases should be handled specially, rather than forcing users to use general techniques "for consistency's sake".
- Printing documents and searching for keywords in documents (find/find-next) are simple functions which users expect.
- People want to see their documents in their original form - WAIStation currently only displays ascii and PICT. This can be fixed with format filters such as Claris' XTND, at the expense of the ability to download arbitrary sections of a document, since such filters require that the document be processed from the beginning.
- Relevance feedback was not obvious - users unfamiliar with the use of relevance feedback did not think to use it - it needs to be made more automatic. One way to do this might be to extend the notion that a question is a conversation, with relevance feedback as context (or body language) - clients or servers can be written that watch their users, and deduce which documents were relevant based on which ones were read. A simpler approach might be to always do relevance feedback, presenting the results in a "see also" list. We tried this, but the Macintosh was too slow to make it useful.
- Communications over 2400 baud modems are too slow to support interactive queries. We found that 9600 baud is barely acceptable, while 56Kb is sufficient to support several users.
- The finder-like interface (drag and drop) is not obvious - Even though the Macintosh Finder is based on drag and drop, no one expected it in an application. Once users were shown what to do, it was very natural. It was also not necessarily the best use of screen space, since it required that both the start and end of the drag be visible on the screen at the same time. Another anomaly worth mentioning is the fact that although we were simulating the finder, we had no "trash can" analogy. Removing a source was accomplished by dragging it onto the desk top and dropping it there, which confused some users.
- The alerting system was crude. For example, there was no visual cue to tell the user that a question had found new documents in the background. Also, the background searches did not exclude previously read documents.
- Headlines often don't give enough context - The headlines displayed in the question window were only about 60 characters long, making it difficult to identify which documents were useful without opening them. Furthermore, there was no provision to display the document's date or the name of the source it came from.

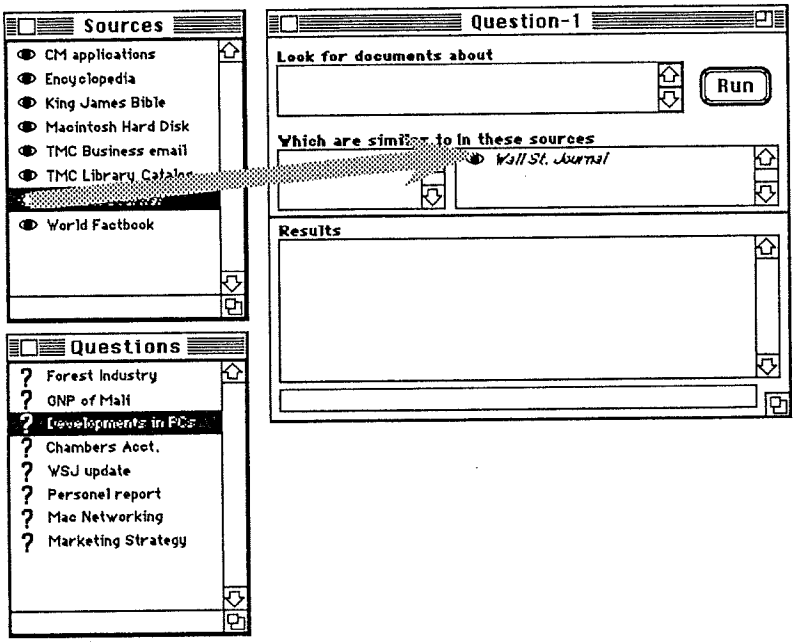

Figure 1 *WAIStation's Sources and Questions windows store the user's personal objects. Dragging a source into a question window specifies that the question will contact the source in order to fulfill its charter.*

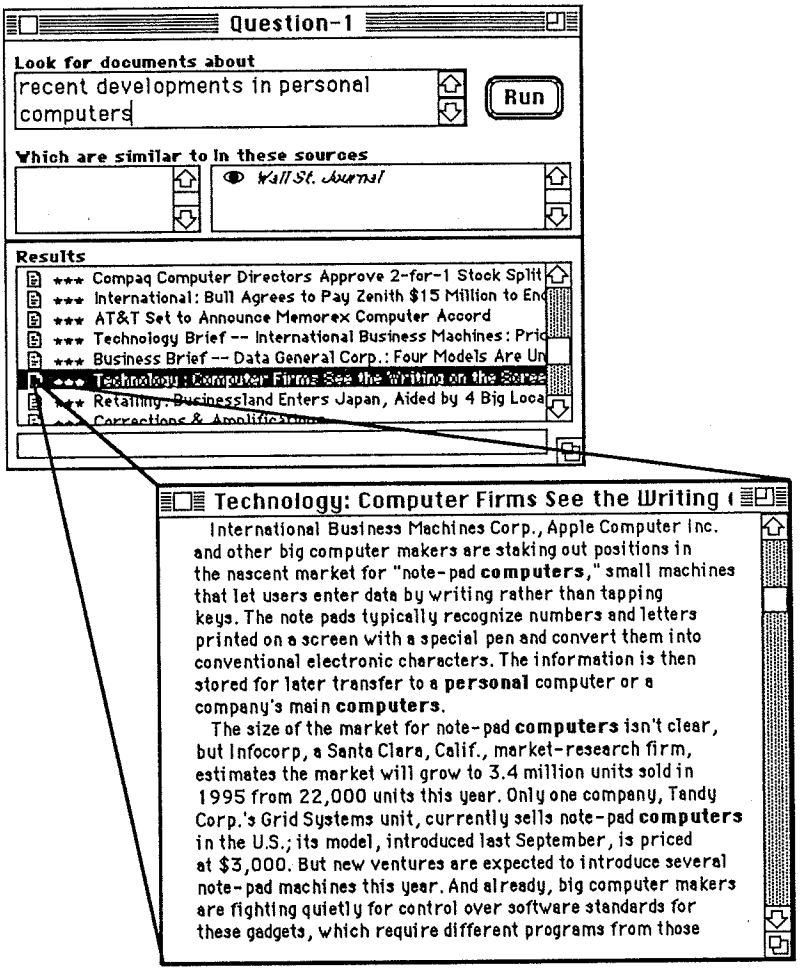

Figure 2 *After running the question, results are displayed in a scrolling list. Double clicking on a result opens a document window. Query words are highlighted.*

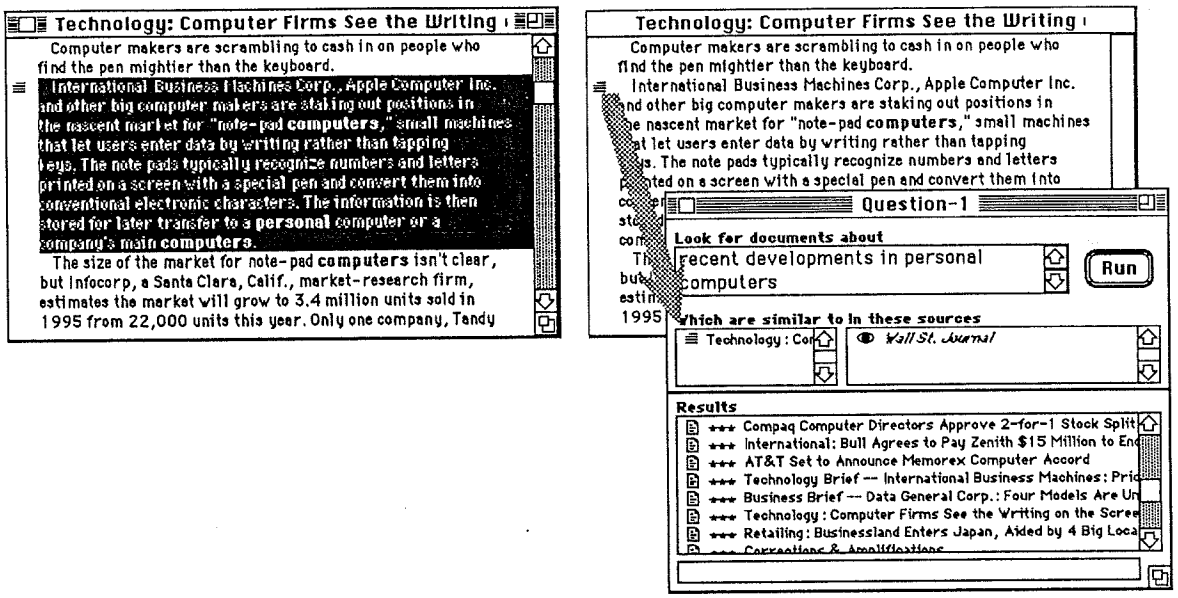

Figure 3 *Relevance feedback is done by selecting a document or part of a document, and dragging the document or paragraph icon into a question.*

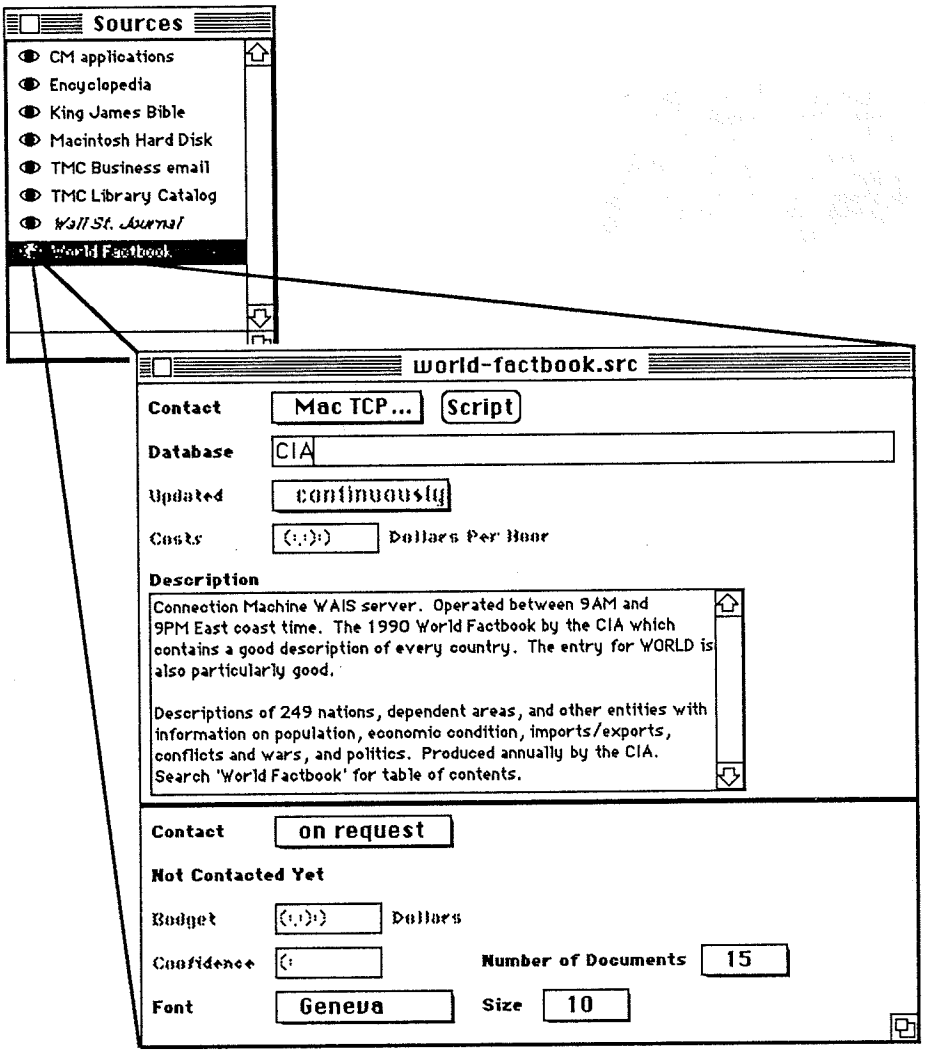

Figure 4 *Double clicking on a source icon opens a source window.*

4. THE ROSEBUD INTERFACE: REPORTERS AND NEWSPAPERS ON THE MACINTOSH

ROSEBUD At A Glance	
Target Machine	Macintosh II, color screen
Effort	5 man-years: 1 man-year HI designer with psychology background, 6 man-months HI designer with engineering background, 6 months graphic designer, 3 man-years programmers.
Number of Users	25
Status	Finished; internal use
Language	Smalltalk, MPW-C
Communications	TCP/IP using IPC package
Designers	Thomas Erickson, Gitta Salomon, Ruth Ritter
Organization	Apple Computer
Availability	Only internally to Apple ATG
Design goals	Serve as research platform for interface and architectural explorations. Allow ordinary users to create personalized information flows; support passive alerting, scanning and capture of information.
Used	Used in various internal tests; not available for the Internet experiment
Problems	Dealing with multiple servers and a directory of servers

Rosebud is a project within Apple Computer's Advanced Technology Group. Its principle objective is to serve as a platform for investigations into what is needed to make remote information accessible to ordinary Macintosh users. The investigations have two foci: human interface components and techniques; and system architecture issues. In this article we focus exclusively on the human interface aspects of Rosebud.

Rosebud is similar to the WAIS interfaces described here in that it uses the Z39.50 protocol to access multiple, remote database; it differs from them in that it contains extra underpinnings for making information access an integral part of the Macintosh environment. The Rosebud system does not currently provide access to the internet WAIS servers (for reasons of network security, rather than basic incompatibilities), and is not available for the internet experiment.

4.1 Design Rationale

The design of the Rosebud interface began with a study of the practices and problems of ordinary information users. The principle focus was on information users at KPMG Peat Marwick in San Jose, the original client site for WAIS; in addition, several groups of users of on-line information services within Apple were also studied (Erickson, 1991). Interviews with accountants at Peat Marwick enabled the designers to put together a schematic of how information (mostly paper-based information) flowed through their offices (figure 5).

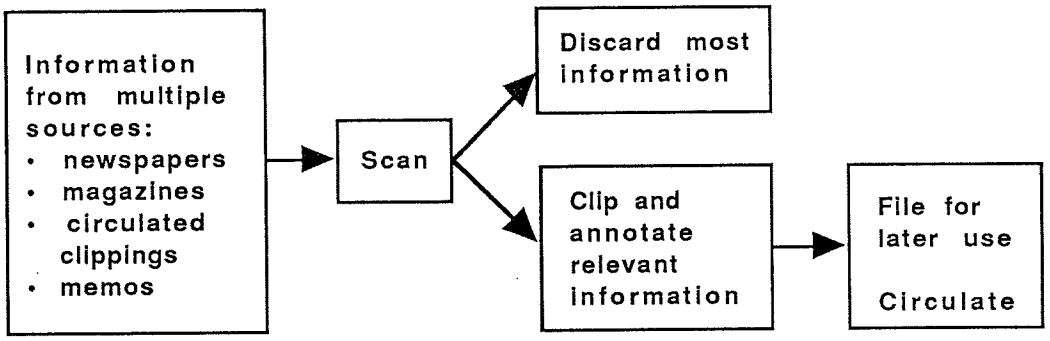

Figure 5 Information flow through accountants' offices.

Several features of this schematic informed the design of Rosebud. First, information typically came to the accountants via newspapers, magazines, and memos; instances where the accountants went out of their way to search for information were less frequent. Second, the accountants never talked about "reading" information; they always spoke of scanning, or skimming it—they didn't have time to read it. This suggested that a good interface should provide a way for the users to scan retrieved information quickly. Third, accountants remarked that they discarded most information, including information that might be useful. Potentially useful information was discarded for two reasons: the accountants didn't have the physical space to store everything, and they knew from experience that if they tried to save too much, they wouldn't be able to find anything later, when they actually needed it. This suggested that giving users access to remote information was just half the problem; users also needed tools for archiving, organizing, and re-retrieving information. Finally, when users did come across information that seemed worth saving, they would typically cut it out (the accountants used, almost exclusively, paper-based information), and then they would annotate it by circling, underlining, or jotting a few notes in the margin. Annotation turned out to be an important concept: not only did it help the user who annotated when the information was re-retrieved later on, but it also helped others scan the information more quickly when copies were passed on to them.

The consequence of these observations was a design for a system which allowed users to define topics of interest which would be automatically retrieved, and would then permit them to scan those items and save them into an environment where they could be annotated, organized and re-retrieved.

4.2 Human Interface

The Rosebud interface design has three components: reporters, newspapers, and notebooks. Reporters are for retrieving information. Users give reporters assignments which specify what to look for, and where to look. This is shown in figure 6: users enter words describing the information in which they're interested, check off the information sources they wish the reporter to search, and, if they so choose, automate the reporter so that it searches the databases on a daily or weekly basis. Upon pressing the "Search" button in the assignment window, a reporter is created, performs the search, and returns with a list of results (figure 7).

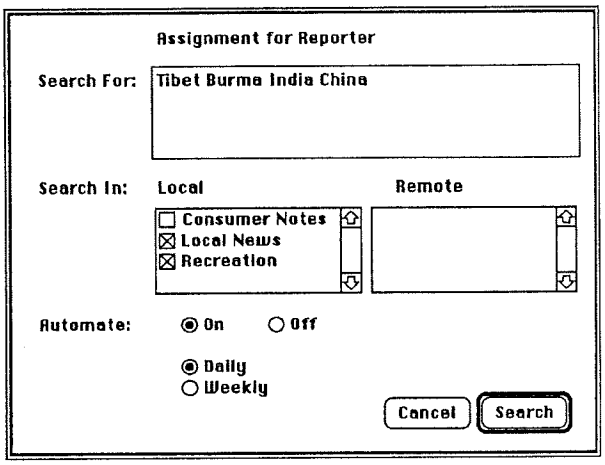

Figure 6 Creating a reporter—the assignment window.

The reporter window (figure 7) provides users with a variety of ways to look over their results, and refine their queries. The results are shown in the "Best Guesses" pane. (The name '"Best Guesses" was chosen to provide some indication that inaccuracy could be expected; our observations of users had shown that they were often mystified by some of the items that showed up as the results of searches.) The asterisks to the left of items indicate their relative relevance, and the pop up menu above the pane allows users to order the list by date or relevance. Simply selecting an item shows a preview of it—a short excerpt with search terms highlighted in boldface (figure 8). Previews are useful because users can get a look at a little bit of the item without incurring the overhead of downloading the whole article over the network. Users also have the options of saving articles to their disks or opening them for viewing. Finally, having looked over their results, users can refine their search in the bottom pane of the window.

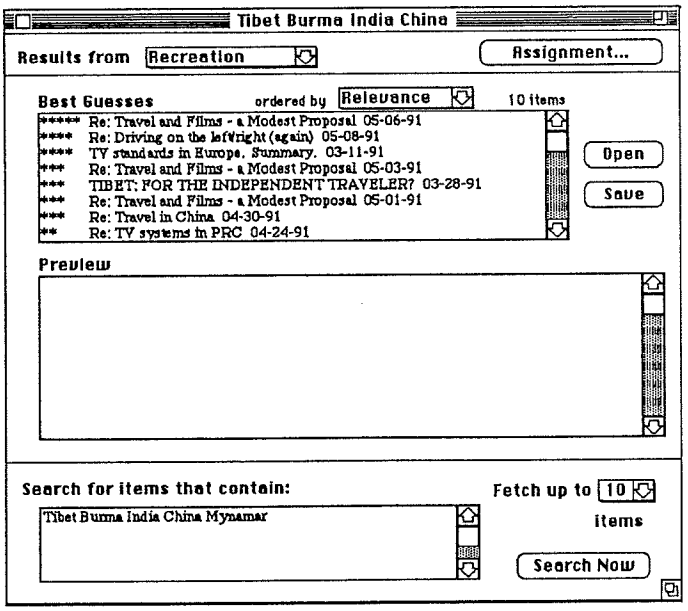

Figure 7 The reporter window contains the results of the search and provides means for previewing, opening, and saving results.

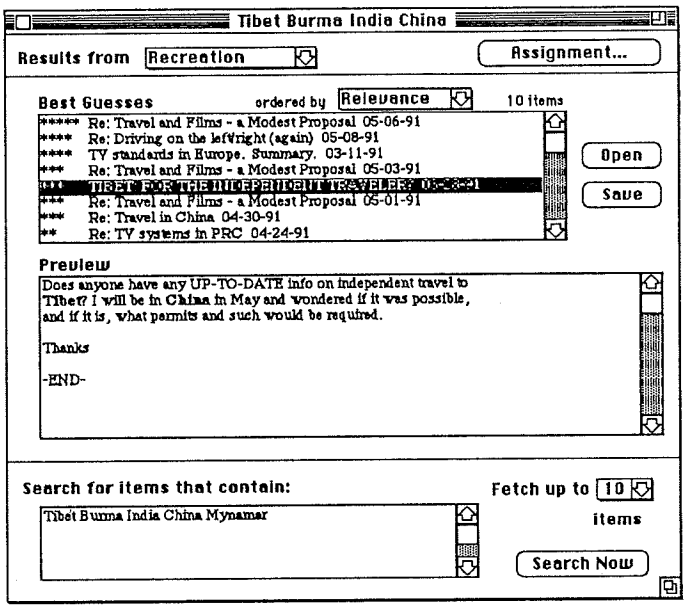

Figure 8 *The reporter window makes it easy to scan through hits. Clicking on a retrieved items generates a preview which shows an excerpt to the hit with the search terms (Tibet and China) highlighted in boldface. The user can refine the query in the lower pane of the window.*

The above sequence occurs whenever a user creates a new reporter. However, since users are likely to use many reporters, and because the initial user studies indicated that ways of skimming through incoming information were important to the accountants, the newspaper was provided to support rapid scanning of new information. The model of a newspaper is quite simple (figure 9): on the left is an index column which contains the names of all reporters, and to the right are two columns of news. Each reporter 'owns' one news column and publishes the title, date and an excerpt of each item in its column. The columns scroll independently, using 'minimalist' scroll bars to prevent the multiple scroll bars from visually overloading the screen. If an excerpt seems interesting, double clicking on it opens the full article in a window, from which it can be viewed, printed, or saved. Thus, rather than having to open up a dozen reporters every morning to see what's new, the user can go to one place, the newspaper.

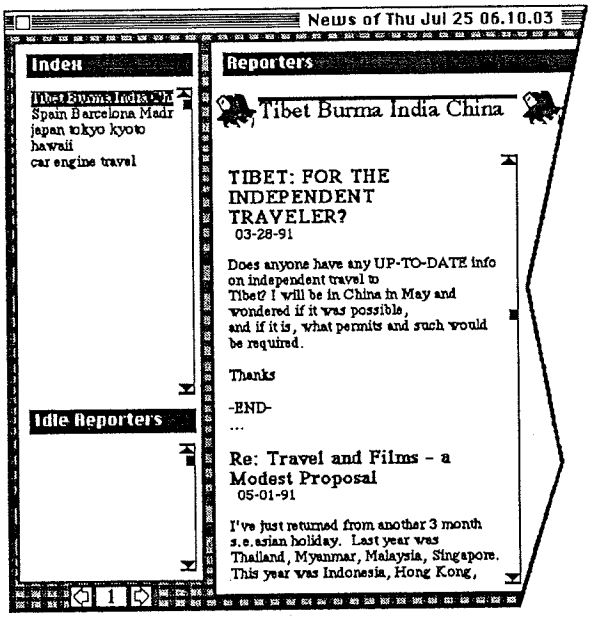

Figure 9 *The newspaper allows users to quickly scan through new items retrieved by the reporters which are working automatically.*

The newspaper can also serve as a control center for the Rosebud interface. The user can open a reporter by clicking on its name or icon at the top of its news column. Consequently, if a reporter's column has strayed from the desired topic, the user can quickly get to the reporter and revise its assignment. The index also lists inactive reporters (those either not automated, or that haven't found anything new since the last newspaper), so they too can be opened, and automated or otherwise adjusted.

A third component of the Rosebud interface—the notebook—was designed but not implemented. Notebooks are environments within which users may save, annotate, and organize retrieved information. Notebooks were designed in response to the observations of Peat Marwick accountants, which indicated the need for an environment which supported the way accountants worked—in particular, notebooks were intended to support annotation, and re-finding retrieved information at a later date. A particularly nice feature of the notebook design was its use of annotations as landmarks for re-finding information. The notebook design, and its rationale, is described in (Erickson, 1991).

4.3 Implementation

Rosebud consists of two parts: a user interface application written in SmallTalk (to facilitate the rapid changes in the interface necessary to effectively conduct interface design research), and a search manager application written in MPW C, which uses TCP/IP and an IPC package to communicate with search engines and remote databases and with the user interface application; like the other WAIS interfaces, Rosebud uses the WAIS protocol package to communicate. The human interface was designed for Macintosh II class machines, with 13 inch color screens. The search manager application runs in the background under MultiFinder, and is able to access information and construct newspapers while the user interface application is not running.

4.4 Observations and Testing Results

The Rosebud human interface was subjected to informal testing on 14 users. Users were told only that Rosebud was an application for finding information, and then given a particular topic to find information on. They were given no help or documentation. Note that although informal, this type of testing is very stringent, in that users approach the application knowing almost nothing about what it is, or why they would actually use it. Data collection consisted simply of recording their questions, observations, and problems as they went along, administering a post-test questionnaire, and then asking them a few, open-ended questions. Here are a few of the more general observations.

- Over 80% of those who tried the Rosebud interface responded very positively to it, and said that they would use something with its capacities as part of their daily work routine. Two thirds of users indicated that they would usually use newspapers to browse through information (instead of reporters).
- At the end of the test, over two thirds of the users said they liked the metaphors of reporters and newspapers; however, almost all users had some difficulty in getting started. The typical problem was that users did not associate reporters with a way of retrieving information. When asked to find information, users first looked for an item called search; when they didn't find this, they usually turned to the newspaper, which is, in fact, where they look for information on a daily basis. It is possible that this problem can be remedied by minor interface changes (e.g. putting a "New Reporter" item in a search menu); alternatively, it may be that the metaphor is inappropriate.
- A number of users were lead astray because they had conceptual models of information retrieval based on their familiarity with query languages and structured databases. Such users tended to be wary of entering search terms because they weren't sure of what the appropriate syntax was, and didn't understand what "relevance" meant. Those that did know what relevance was wanted to know how the information server calculated it.
- Users liked previews a lot—especially the feature of highlighting keywords in boldface. They wanted to see boldface keywords in the newspaper and article windows. Users also wanted the ability to select text in the newspaper and article windows and change the style or font themselves, so that they could annotate significant items. This parallels practices observed in our initial observations of accountants, where we found that annotation plays several important roles.
- A variety of low level interface problems, due to terminology or graphic design were discovered. Some examples: users did not usually recognize the asterisks in the "Best Guesses" window as indicators of relevance; users didn't think that "idle reporters" was a good name, and said that it was very important to distinguish between reporters which had found nothing, and those which weren't looking.

As of this writing, the next phase of Rosebud human interface testing is about to begin. In this phase, a small set of users will be observed over the course of a month, in which they have the option of using Rosebud from their desktop machines to access meaningful data. This phase of testing will allow a more realistic assessment of Rosebud, in that it will last long enough to permit users to build up their own set of reporters, and to access newspapers which contain information of personal import.

5. X WINDOWS BASED INTERFACE FOR WAIS: XWAIS

XWAIS At A Glance	
Target Machine	X-windows terminals on unix machines
Effort	4 man-months
Number of Users	500
Status	finished, freely distributed
Language	C
Communications	TCP/IP
Designer	Jonathan Goldman
Organization	Thinking Machines
Availability	Available anonymous FTP from /public/wais/wais*.tar.Z@think.com
Design goals	Copy WAIStation so that we can leverage one design, portable and based only on freeware Display data in many different formats (image, text, etc)
Used	Used in the internet experiment Heavy use by X users within Thinking Machines and outside
Problems	Installing it has caused many users to stumble. The number of variables (architectures, X directory structures) makes it difficult to make it portable touch on the ability to handle different types (this is unique to this interface). uses other programs to help (like interapplication communication)

The WAIS interface for the X Windows environment was developed for the Internet experiment to provide an X Windows based interface for a growing community. It was built to look as much like the Macintosh WAIS interface (WAIStation) as possible, given the limitations of the freely distributed X Windows software. Since the metaphors in XWAIS are nearly the same as those for WAIStation, a user of one system can easily move to the other, without having to learn much new. In fact, the underlying data structures are identical to those in WAIStation, so questions can be copied from a Macintosh to a UNIX machine running XWAIS, and used without modification.

XWAIS supports interactive WAIS access, including question entering, source selection, addition of relevant documents and pieces of documents. Unlike WAIStation, XWAIS retrieves an entire document when requested, instead of just the parts being viewed. We decided this was acceptable, since the underlying networks for X will most likely be fast.

Since XWAIS runs under X windows, and was built for the UNIX operating system, it can take advantage of the tools available for these systems to display a wide range of document formats. A simple filter interface is provided in the application (as an X resource) to allow a user to select the tool required for a given type of document, e.g, if the document is a postscript file, xps can be used to view it. This is a feature that is not available in any of the other user interfaces described here.

In order to distribute this software without restriction, XWAIS uses the freely distributed Athena Widget set included in the X11R4 release from MIT. Although these widgets don't look as nice as some others that are available, they can be used to build a useful interface. Some aspects of this interface are restricted by the nature of the widgets available. XWAIS was built using the Xt X Toolkit Intrinsics, and allows a large amount of customization of the appearance of the display using X resources. The application relies heavily on the Xt resource mechanism, and will not run unless these resources are in place. The "object-oriented" feel of these widgets made building the interface rather easy, once the widget with the closest desired functionality was found. Finding the correct widget was the hardest part. Most of the actual behavior of the interface is controlled by "call-backs" - the methods that widgets inherit.

The XWAIS application is actually two separate applications: XWAIS, a simple shell for selecting sources and questions, and xwaisq, the application that actually performs WAIS transactions. The C code in xwaisq is also used

in waisq, the shell-support program for GNU Emacs WAIS. This allows users to use simple UNIX facilities to submit questions created by xwaisq using waisq (e.g. a crontab entry to periodically query a server).

The implementation for XWAIS was done in C (6k lines), using the X11R4 release of X windows from MIT, the Xt X Toolkit Intrinsics, and the Athena Widget Set, included in the X Windows release.

XWAIS is a text-based user interface built in a graphical window environment. Some additional graphical metaphors would be desirable, but the limited widget sets precluded that. It would take a considerably larger amount of work to add much graphics to this application. Perhaps some other X toolkit would provide simpler methods for doing this.

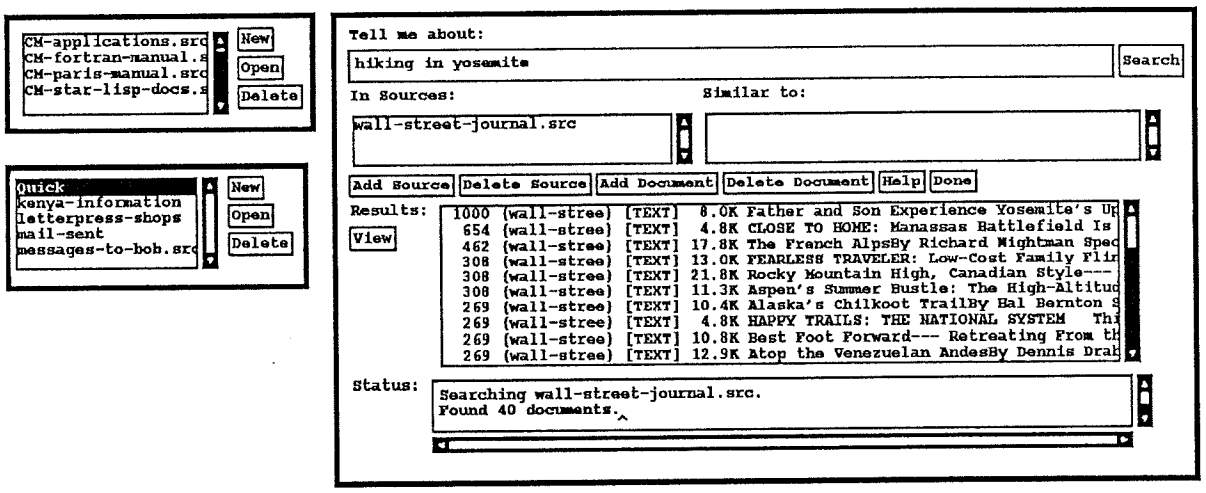

Figure 10 The XWAIS interface, including the Questions and Sources windows, and an open question.

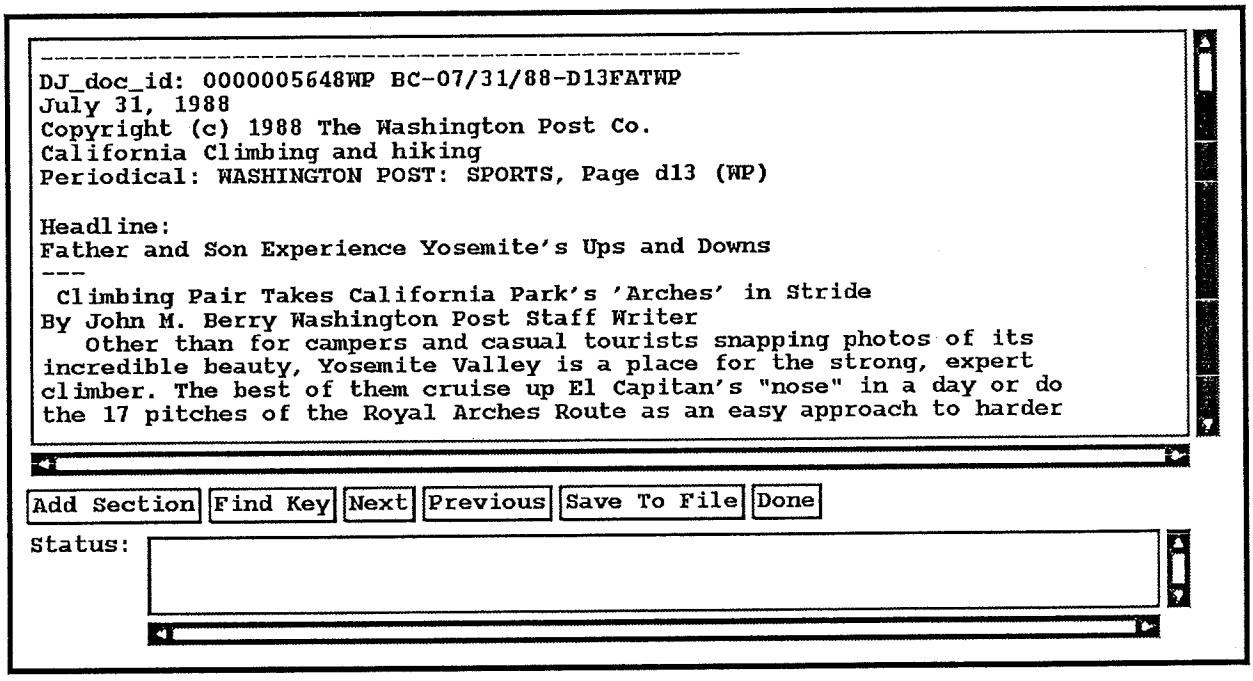

Figure 11 A document displayed in the XWAIS interface.

6. GNU EMACS WAIS INTERFACE: GWAIS

GWAIS At A Glance	
Target Machine	terminals on unix machines
Effort	2 man-months
Number of Users	500
Status	finished, freely distributed
Language	gnu-lisp, and C
Communications	TCP/IP
Designer	Jonathan Goldman
Organization	Thinking Machines
Availability	Available anonymous FTP from /public/wais/wais*.tar.Z@think.com
Design goals	Copy WAIStation so that we can leverage one design, Use precedent from other gnu-emacs applications: RMAIL, dired
Used	Used in the Internet experiment with heavy use by some gnu-emacs users
Problems	Dealing with the directory of servers. Using passive alerting

The WAIS interface on GNU-Emacs/Unix (GNU) was developed specifically for the Internet experiment for a technically strong user population. The reasons it was developed were: the large number of emacs users, the extensibility, the ubiquitous nature of character display terminals, and the component nature of emacs which meant WAIS could be integrated into email, bboards, and programming tools.

The design of the interface was a cross between WAIStation and other emacs interfaces. The direct manipulation of WAIStation was replaced by command keys, as is common in emacs applications. The choice of command keys were modeled on the dired and RMAIL emacs applications.

GWAIS allows users to access the interactive features of WAIS: question entering, relevance feedback, displaying document, and source selection. An extra feature, not found in the other interfaces, is an interface to an indexer for creating sources, but it appears that this feature is not heavily used. Furthermore it allows questions to be saved, but it depends on the user to automate the update of questions and sources using cron or other Unix tools. Graphic documents can be displayed on X Windows terminals if the user has set up the environment variables.

The implementation of GWAIS was in emacs lisp (2K lines) and in C code (3K lines). About half of the time of a typical search and retrieval is spent in reading the data into lisp.

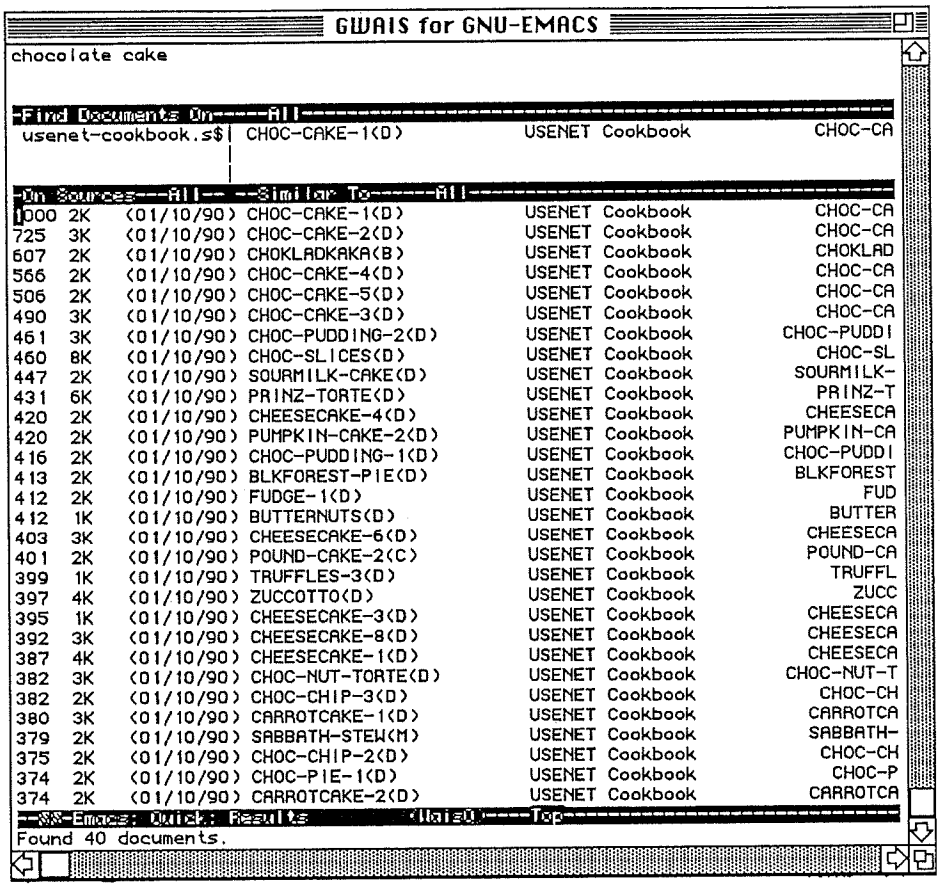

Figure 12 The GWAIS interface, displaying the results of a relevance feedback search.

7. SCREEN BASED (TERMINAL) WAIS INTERFACE: SWAIS

SWAIS At A Glance	
Target Machine	Terminals connected to Unix systems
Effort	1 man-month
Number of Users	900
Status	beta
Language	C
Communications	TCP/IP
Designer	John Curran
Organization	NSF Network Service Center
Availability	To be included in WAIS release, anonymous FTP from /public/wais/wais*.tar.Z@think.com
Design goals	Highly Portable, Provide straight-forward user interface, Utilize existing application key mappings (rn, vi, emacs), Support multiple servers per query, Allow for personal "source" directory and a common source directory, Allow for useful source discovery via searches, Provide simple active tool with little state (no question storage, relevance feedback, or passive notification)
Used	Internet users via telnet: k-12 students, educators, user services staff, librarians, and (occasionally) network staff
Problems	Dealing with the directory of servers. Lack of information in many server-returned records. Providing simple and uniform nomenclature Planning for large numbers of sources.

To open WAIS to a wider community of users, an interface was developed to run on dumb terminals or over telnet sessions. It is called "SWAIS" for Screen WAIS since it is uses a character display terminal screen for the interface. The user communities that this interface can serve are dial-in users, telnet users, and low-end terminal users.

The design of the interface involved 3 screens: a single screen listing all known servers that the user could pick from; a list of search result documents headlines; and a document display screen. Listing all servers and allowing users to pick which servers to use encourages users to ask questions of multiple servers. Unlike the other interfaces, the sources list shows what site runs it and how much it costs (if anything). The resulting document screen includes headlines and how many lines it is, but its innovation is to show what source it came from.

It does not handle relevance feedback or downloading new sources from the directory of servers. Another drawback is using it with large numbers of sources since moving around the list requires scrolling. On the other hand, this server has proven to be very popular on the Internet because of its ease of use, all a user has to do is telnet to a specific machine to use it.

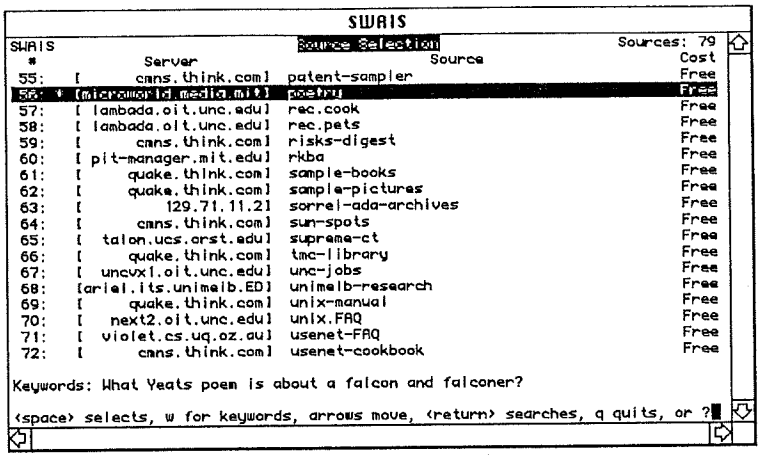

Figure 13 *The SWAIS query building screen. The poetry source is selected, and search terms are entered. This interface does not currently support relevance feedback.*

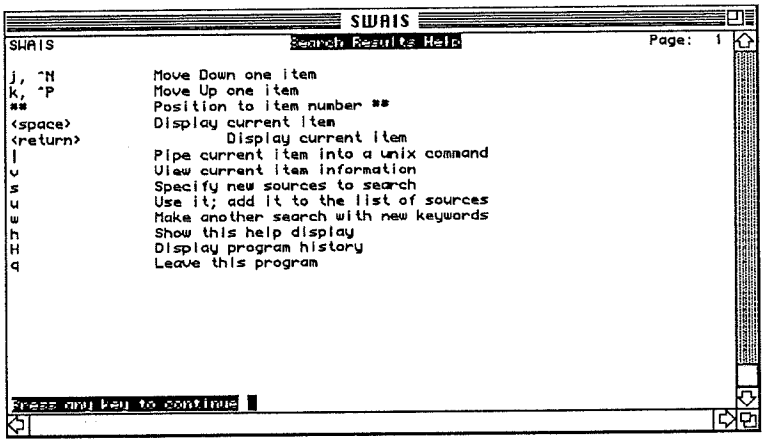

Figure 14 *The SWAIS help screen.*

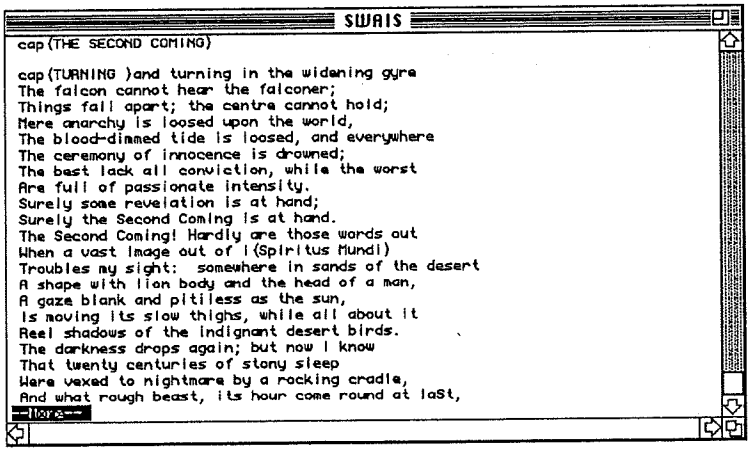

Figure 15 *A document displayed in SWAIS.*

8. CONCLUSION

This paper has described five interfaces developed in the context of the Wide Area Information Servers project. The interfaces presented here were developed with different constraints in mind, so it is not useful to compare them directly; instead they may serve as examples of differing responses to issues such as screen size, workstation power and intelligence, communication speeds, and user needs and practices.

The interfaces designed so far have addressed some of the critical issues for end-users to accomplish interactive searches in a wide area network. These include ways of finding which information servers contain relevant information, supporting searching by ordinary users, and supporting browsing of, and passive alerting about, newly retrieved information. The alerting aspects of the interfaces have not been tested much in this environment due to the lack of appropriate data sources for this type of searching. It is probably fair to say that any of the design solutions described here can be improved upon by further work.

The WAIS Internet experiment has revealed a number of issues requiring further work. In the Internet environment we have observed (in the logs of user queries) that users have a difficult time finding out what is in a database, thus demonstrating that there is a lack of browsing or scanning facilities in the interfaces, protocol, and servers, as well as a general shortage of descriptive information about databases.

Finally, there are a variety of other issues raised during the studies of the Peat Marwick accountants which have received little or no work. Document layout is one such problem. Accountants mentioned that sometimes they want to retrieve documents not because of the information they contain, but to look at their layouts (accountants will often examine successful proposals to a client when preparing a new proposal). More generally, users regard pictures, diagrams, tables, and charts as essential components of a document's content. Unfortunately, support for different document formats, and for the retrieval and display of non-textual information within them is very limited on most existing clients.

Another issue is called the boilerplate problem. Accounting documents often contain a large amount of boilerplate, standard text which varies little from document to document. What tools are needed to allow users to effectively retrieve, order, and browse a large set of documents which are 95% similar? Note that boilerplate is characteristic of a wide variety of business proposals and legal documents, not just accounting documents. In fact, the analog to boilerplate occurs in scientific documents in which standard terms and descriptions are used to describe procedures and methods used in an investigation.

A number of other issues remain to be addressed. Users are very interested in being able to see what queries other users are conducting, and what information servers and articles are most popular. A frequent suggestion is to allow users to rate the 'goodness' of articles they retrieve. However, in a commercial setting, information about the kind of questions being posed by a particular company or person can be revealing and valuable. Clearly, the utility that such information could provide must be balanced by concerns about confidentiality and privacy, and mechanisms for user control of descriptive information are essential. Other issues include how to control the pricing, copyright, and distribution issues which accompany 'for-pay' information.

In summary, there is an immense amount of work to be done. A central part of this work involves further research and development of interfaces. We hope that designers will find that WAIS—with its common protocol and defined infrastructure—can serve as a platform from which to pursue these, and other, research issues.

FOR MORE INFORMATION ON WAIS

The success of a WAIS-like system depends on a critical mass of users and information services. In order to encourage development and use, Thinking Machines is making the source code for a WAIS protocol implementation freely available. While this software is available at no cost, it comes with no support. We hope that it will facilitate others in developing servers and clients.

For more information, please contact:

Barbara Lincoln (barbara@think.com)

Thinking Machines Corporation
1010 El Camino Real, Suite 310 245 First Street

Menlo Park, CA 94025
415-329-9300

Cambridge, MA 02142
617-234-1000

REFERENCES

(Davis, 1990) F. Davis, B. Kahle, H. Morris, J. Salem, T. Shen, R. Wang, J. Sui, and M. Grinbaum, "WAIS Interface Protocol Functional Specification," Thinking Machines Corporation, April 1990. Available via anonymous ftp: /pub/wais/doc/wais-concepts.txt@quake.think.com or wais server wais-docs.src

(Dongarra, 1987) J. Dongarra and E. Grosse, "Distribution of Mathematical Software Via Electronic Mail," CACM, 1987, Vol. 30, pp. 403-407.

(Egan, 1989) D. Egan, J. Remde, L. Gomez, T. Landauer, J. Eberhardt, and C. Lochbaum, "Formative Design-Evaluation of SuperBook," ACM Transactions on Office Information Systems, January 1989.

(Erickson, 1991) T. Erickson and G. Salomon, "Designing a Desktop Information System: Observations and Issues," Proceedings of the ACM Human Computer Interaction Conference, 1991. ACM Press, April 1991, pp. 49-54.

(Ginther-Webster, 1990) K. Ginther-Webster, "Project Mercury," AI Magazine, 1990, pp. 25-26.

(GNU) For information on the Free Software Foundation and the GNU project, see: /pub/gnu/*@prep.ai.mit.edu. For emacs see: /pub/gnu/emacs-*.tar.Z@prep.ai.mit.edu.

(Kahle, 1991a) B. Kahle and A. Medlar, "An Information System for Corporate Users: Wide Area Information Servers," April 1991. Online Magazine, September 1991.

(Kahle, 1991b) B. Kahle, J. Goldman, H. Morris, T. Shen, "Electronic Publishing Experiment on the Internet - Wide Area Information Servers" In preparation.

(Malone, 1986) T. Malone, K. Grant, and F. Turback, "The Information Lens: An Intelligent System for Information Sharing in Organizations; In Human Factors In Computing Systems," CHI'86 Conference Proceedings, Boston, MA; ACM, New York, 1986, pp. 1-8.

(Mosis) For more information on the Mosis fabrication facility write to mosis@mosis.edu.

(NeXT) NeXT Computer Inc., 900 Chesapeake, Redwood City, California, 94063, (415) 366-0900.

(NISO, 88) "Z39.50-1988: Information Retrieval Service Definition and Protocol Specification for Library Applications," National Information Standards Organization (Z39), P.O. Box 1056, Bethesda, MD 20817. (301) 975-2814. Available from Document Center, Belmont, CA. Telephone (415) 591-7600.

(ON Technology) ON Technology Inc., 155 Second Street, Cambridge, Massachusetts, 02141, (617) 876-0900.

(VERITY) Verity Inc., 1550 Plymouth Street, Mountain View, California, 94043, (415) 960-7600.

ACKNOWLEDGMENTS

The Rosebud interface was designed by Gitta Salomon, Tom Erickson, and Ruth Ritter. Kevin Tiene had significant impact on the interface design, and also implemented the whole thing. Kevin Tiene, Charlie Bedard, David Casseres, and Eric Roth designed and implemented the search manager, and other underpinnings of Rosebud. Steve Cisler and Janet Vratny-Watts, of the Apple Library, provided valuable information on the state of the on-line information universe, and on the needs of information end users.

At Thinking Machines: Ottavia Bassetti, Franklin Davis, Patrick Bray, Danny Hillis, Rob Jones, Barbara Lincoln, Gordon Linoff, Chris Madsen, Gary Rancourt, Sandy Raymond, Tracy Shen, Craig Stanfill, Steve Schwartz, Robert Thau, Ephraim Vishniac, David Waltz, and Uri Wilensky.

At the National Science Foundation: Suzi, Alex, et. al.